Docket No : **POU920000194US1**

Inventor : **GRECH, et al.**
Title : **METHOD AND SYSTEM FOR ISOLATING AND SIMULATING DROPPED PACKETS IN A COMPUTER NETWORK**

APPLICATION FOR UNITED STATES

LETTERS PATENT

"Express Mail" Mailing Label No.: **ET089965573US**
Date of Deposit: **November 28, 2001**

I hereby certify that this paper is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Box Patent Application, U.S. Patent & Trademark Office, P.O. Box 2327, Arlington, VA 22202.

Name: **Ann S. Lund**

Signature: Ann S. Lund

METHOD AND SYSTEM FOR ISOLATING AND SIMULATING DROPPED PACKETS IN A COMPUTER NETWORK

FIELD OF THE INVENTION

[0001] The present invention relates generally to computer networks. More particularly, the present invention relates to diagnosing the source of network errors by providing an automated tool to isolate and simulate dropped packets in a computer network.

BACKGROUND OF THE INVENTION

[0002] The ability to rapidly identify and fix transmission errors in network systems has become increasingly important as more businesses have come to rely on applications and data that span multiple networks. Transmission packets in a computer network can be dropped at multiple points in the network resulting in applications that do not respond, experience poor performance as a result of packet re-transmits, or behave unexpectedly. Dropped packets can be caused by a failing router or communication controller hardware, as well as by software defects. Isolating dropped packets requires traces to be gathered at multiple points in a network. Once collected, these traces must be correlated and analyzed by multiple experts as dictated by virtue of the various computing and network environments involved. This correlation and analysis process is manual and time consuming, which can result in long and costly problem isolation scenarios.

[0003] For example, a mainframe application may not be responding due to a partner application that did not respond, resulting in packets that did not reach their destination. To determine why and where packets were lost, traces would need to be collected on the mainframe, on transactions leaving the mainframe, at retransmission points, and at the target destination. Multiple experts would be required to analyze the packet traces and determine at which point packets were dropped. In another example, a mainframe application, using a mainframe operating system (e.g. OS/390) may stop

functioning while communicating with an engineering workstation (e.g. RS/6000) application. Here, the packet flow could include going from the mainframe to a controller (e.g. 3174) then over a token ring LAN (local area network) to reach the engineering workstation. The network protocol could be any one known in the art including SNA (System Network Architecture) and IP (Internet Protocol). Experts would be required on the mainframe, on the controller, on the token ring LAN, and on the engineering workstation in order to fully analyze the problem. In addition, multiple recreations of the network failure would be required to diagnose the problem.

SUMMARY OF THE INVENTION

[0004] An exemplary embodiment of the present invention is a method and system for isolating dropped packets in a computer network. A request for network analysis that includes a source node and a destination node is received by the invention. A map of the expected path between the two points, including the probes along the route is then generated. A capture filter profile for each probe along the route is created. A request to perform data collection is transmitted, along with the capture filter profile, to each of the probes along the route. Data is received back from the data collection in the form of a data log. Exception data is generated by comparing the data log to the expected path between the two points. Additional embodiments include a system and storage medium for isolating dropped packets in a computer network.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIG. 1 is a block diagram of an exemplary embodiment of a system for isolating dropped packets in a computer network.

[0006] FIG. 2 is a flowchart of an exemplary high level application flow of the software for performing isolation of dropped packets in a computer network.

[0007] FIG. 3 is an exemplary collector application flowchart.

[0008] FIG. 4 is an exemplary collection task application flowchart.

[0009] FIG. 5 is an exemplary correlator application data flow diagram.

DETAILED DESCRIPTION OF THE INVENTION

[0010] The present invention aids in diagnosing network problems and helps to minimize the amount of human expertise required to debug network problems by providing an automated mechanism to isolate and simulate dropped packets. A block diagram of an exemplary embodiment of a system to carry out the present invention is shown in FIG. 1. The block diagram depicts an engineering workstation (e.g. RS/6000) source system 102 and a mainframe (e.g. S/390) destination system 104. In FIG. 1 the source system 102 and destination system 104 are separated by various networks 116 that are connected by gateways 114. The source system 102 and destination system 104 can be built on any type of computer system hardware that is known in the art. The networks 116 can be based on any communication protocol that is known in the art (e.g. SNA and IP). FIG. 1 also depicts the destination system 104 being connected to the network 116 through a controller 118. Probes 106 have been inserted at various points in the network to perform data collection. The probes 106 can be put into the network ahead of time or be inserted in response to a specific network problem. In addition, the probes 106 could be collecting data on a continuous basis about each packet that passes through the probe 106 or collecting data only in response to a request for problem analysis of a specific network problem. These probes 106 are connected to the problem isolation system 108 that drives the collection and correlation of the probe 106 data.

[0011] Many system configurations are possible, the block diagram in FIG. 1 is meant to illustrate one of these configuration and not meant to limit the present invention to this specific configuration. Any network configurations, both hardware and software, that are capable of supporting probes can be used with the present invention. The concept of the present invention described herein is extensible to a variety of physical transport mediums including twisted pair cable, coax and fiber. The present invention is also extensible to a variety of link layer protocols including ethernet, token ring and ESCON.

[0012] Probes have been developed for most major media types including token ring, ethernet, fast ethernet, asynchronous transfer mode (ATM), frame relay, and fiber distributed data interface (FDDI). Depending on the media type, probes have been designed to be connected to hubs and rings, or integrated into switch backplanes. As is known in the art, many probes support both network statistics and capture functions. Referring to FIG. 1, probes 106 have been inserted at various points in the network 116 to identify the location at which the packets are lost. These probes 106 are linked to the problem isolation system 108 and each probe 106 will log information depending on the setting of a capture filter by the problem isolation system 108. The capture filter of each probe 106 will be programmed by the problem isolation system 108 at the beginning of each analysis event. A probe 106 will only log packets that satisfy the capture filter. During analysis the problem isolation system 108 will request information from each probe 106.

[0013] The connectivity from probe 106 to the problem isolation system 108 can be achieved using industry standard protocols including Transmission Control Protocol/Internet Protocol (TCP/IP) over a network infrastructure. Connectivity via an existing network at a corporation is an option because in an exemplary embodiment the filtering of the packets will occur at the probes 106 during diagnosis, and communication between probes 106 and the problem isolation system 108 of the present invention can be easily distinguished from regular network traffic. Further, because the probes 106 are capable of buffering captured data, no additional traffic need be generated during the measurement interval. In another embodiment, a separate network can be utilized to connect the probes 106 to the problem isolation system 108. This would increase the cost but would provide some amount of additional reliability and separation.

[0014] FIG. 2 is a flowchart of an exemplary high level application flow of the problem isolation system 108 for performing the present invention. An analysis request 202 initiates the process. In an exemplary embodiment, the analysis request 202 contains a source node, a destination node and a network protocol identifier. An analysis request

202 can be initiated manually by an operator using the collector application 300. An analysis request 202 could also be initiated automatically by an agent in a network such as a client or server. To facilitate automatic initiation of problem analysis, a protocol stack can invoke a service in the agent to send an analysis request 202 to the collector application 300. In an exemplary embodiment, the message sent to the collector application 300 would include the source or originator, the destination, information about the network protocol, and optionally, any information that the originator has about the route between the two endpoints. An endpoint may be programmed to request analysis when a retransmission threshold has been exceeded, or based on some other criteria.

[0015] When the collector application 300 receives an analysis request 202 it will perform some preliminary actions and then schedule the collector task application 400. The scheduling of the collector task application 400 can be restricted by some user programmed criteria. For example, it might be that a field engineer wants to focus on failures between a small number of selected endpoints. The collector task application 400 could then be programmed to reject all requests for analysis between other network endpoints. In an exemplary embodiment, when the collector task application 400 is complete, the correlator application 500 will be invoked to perform analysis to determine at which point the packets were lost and to output this information in the form of exception data 204. Each of these application modules is discussed in more detail below.

[0016] FIG. 3 is a flowchart of processing performed by an exemplary collector application 300. At step 302, the analysis request 202 containing a source 102, a destination 104, and a network protocol type is received by the collector application 300. Next, at step 304, the set of possible routes between the source 102 and destination 104 are mapped, along with the probes required for data capture along these routes. Generally this set contains one route but in some cases there may be multiple routes. The set of possible routes can be determined by topology information accessed by the collector application 300; by routing and protocol information provided in the analysis request 202; and by information provided by elements in the network (e.g. a router table). If a route is

not found, the software at this step can determine a route between the source 102 and destination 104 by observing frames injected by probes 106 in the network. Once a route is found, the smallest set of probes required to capture data along the route is determined.

[0017] At step 306, for each probe 106 in the set of probes determined in step 304, a capture filter is created. The capture filter is formulated to be as restrictive as possible in capturing data between the two endpoints. The capture filter can be determined by information provided in the analysis request, such as source 102 and destination 104 address information, and by the protocol used to communicate between the two endpoints. Additionally, the position of a probe in the network and the type of network segment that the probe is attached to can be used to determine the capture filter. A provision can be made in the analysis request message 202 to allow the originator to suggest more detail to the collector application 300 about the capture filter that should be programmed. For example, if frequent re-transmissions are occurring on a reliable link that has been established between the two endpoints, information could be passed in the analysis request message 202 that identifies the reliable link, and a more restrictive filter could be programmed. For example, if the two endpoints are using an 802.2 protocol to communicate in a LAN environment, such as SNA, in addition to the source and destination MAC addresses, the originator could include the source and destination Service Access Point (SAP's), thereby uniquely identifying a specific 802.2 connection.

[0018] At step 308, after the capture filters for each probe 106 are determined, they are programmed into each probe 106, and each probe 106 is instructed to begin capturing data. The probes 106 are also instructed on the duration of the capture. The duration of a capture may be determined by a combination of a period of time that is suggested by the originating end point in the analysis request message; the capacity of the probe capture buffer; and the collector application 300 imposed scheduling constraints. Steps 306 and 308 are performed for each probe 106 in the set of probes determined in step 304.

[0019] Sample pseudo code for an exemplary collector application 300 is as follows:

```
Main:
        Wait for an Analysis_Request;
        Schedule_Execution;
        Return to Main.

Schedule_Execution:
        Check if there are any restrictions on the route;
        Determine route and probes to use;
        For each probe:
        Create capture filter profile that includes source and destination
        address of the packets to be collected as well as the network
        protocol used;
                Spawn Collection_Task;
        Return.
```

[0020] FIG. 4 is a flowchart of processing performed by an exemplary collection task application 400. This collection task application 400 is called for each probe 106 in the set of probes as determined by the collector application 300. At step 402, the collection task 400 waits for the time specified as the duration of the capture to expire. Next, at step 404, a unique packet identifier is created for each packet received by the probe 106. In an exemplary embodiment, the unique packet identifier includes the source 102, destination 104, probe identifier, and a unique identifier. The unique identifier is determined based on the network protocol associated with the packet. The packets can be multi-protocol packets, including SNA and IP packets as indicated by the frame type indicator. At step 406, the application checks to see if the packet identifier matches the type of packet specified by the capture filter profile for the probe. If it does match, a log entry containing the unique packet identifier is created. Steps 404 and 406 are performed for each packet received by the probe 106.

[0021] Sample pseudo code for an exemplary collection task application 400 is as follows:

[0022] Collection_Task:

When timer expires request data from probes;
For each packet received by the probe do:
Determine protocol of packet;
Based on protocol retrieve unique identification information for packet;
Use unique identification information to form a packet identifier which also includes the probe on which the packet was seen;
Write packet identifier to the log if it matches the filter profile established previously;
End;
End of Collection Task.

[0023] FIG. 5 depicts the data flow of an exemplary correlator application 500. The input data includes the list of expected probes between the source and destination 502 created in step 304 of FIG. 3 and the log entries 504 created in step 406 of FIG. 4. The output data includes exception data 204 that can be used to isolate the dropped packet. This exception data 204 can be used to initiate a software program or to trigger a hardware action. For example, it could be used to trigger an automatic data re-transmit at a particular node or it could be used to generate a notice to be sent a particular person notifying them of the error. Based on the data collected by the collector application 300 and the collector task 400 many kinds of analysis can be performed in order to isolate the dropped packet in the network.

[0024] In an exemplary embodiment of the correlator application 500, exception data can be created without collecting the contents of the probe capture buffers. Two separate frame counts can be determined for each probe 106 in the measurement set for a given time interval. One count can indicate the number of frames sent from a source probe 106 and the other count can indicate the number of frames sent from a destination probe 106. If there is a statistically significant difference in the number of packets observed between the respective counts of two probes 106 along a route during the specified time interval, then a failing network element can be identified. Statistical significance can be based on the total number of packets transmitted by the source

destination probes, and the error in synchronization of the sample interval for each of the probes in relation to the duration of the sample interval. In an exemplary embodiment, the time interval would be made equal to the entire capture interval for maximum accuracy.

[0025] The next step in creating exception data, in an exemplary embodiment, would be conducted if the network protocol adds sequence numbers to the frames in the packets. If a protocol is executed that includes sequence numbers, an exemplary embodiment of the correlator application 500 would start by looking at the data from the probe 106 closest to the destination 104 for gaps in the sequence numbers. If a gap in the sequence numbers is present, then it indicates at least one dropped packet. An exemplary embodiment of the correlator application 500 would then locate the packet with the highest sequence number prior to the gap and proceed backwards from the probe 106 along all routes extending from the segment that contains the probe 106. It would then identify this packet on the prior segment to see if a corresponding gap exists on the previous segments. If the gap doesn't exist on a previous segment than an element has been identified which is dropping packets.

[0026] Using the data collected by the problem isolation system 108, there are many additional methods of looking for lost packets and determining the location of a failing component. In exemplary embodiment, exception data can be created by viewing each packet generated by the source 102 and an attempt could be made to trace the path of each packet through the network. If sequence numbers are present, and allowing for latency of network elements, the correlator application 500 could look for the corresponding element on the output side of the network element. Similarly, the correlator application 500 could do correlation of packets without sequence numbers based on assumptions about propagation delay through the network elements along a route. When the analysis is complete, the exception data 204 could include a map of the topology which indicates the network elements that were identified as dropping packets. If the above analysis shows no problem in the network, further analysis of captured data can be performed to determine which end point is failing. This can be made based on a

knowledge of the protocol being exercised and request response pairs. If the destination is not producing responses then it is in error, and if the source is receiving responses but still requesting retransmission of packets then the source is implicated.

[0027] Sample pseudo code for an exemplary correlator application 506 is as follows:

```
Main:
        Create packet list to keep track of what probes saw each unique packet;
        Spawn Exception_Task;
        Do forever:
                For each packet in the log:
                        If packet entry does not exist in packet list add packet entry
                        for packet to the packet list;
                        On initial create of the packet entry add the source and
                        destination address of the packet to the entry;
                        Add probe identifier to packet entry if not already entered
                        for that probe;
        End;
End.

Exception_Task:
        Do at specified intervals:
                For each packet entry in the packet list:
                        Identify those entries that are missing probe identifiers and
                        write out packet entry including probe list for human
                        analysis;
End.
```

[0028] The present invention can be extended to encompass more than a single two point route and is extensible to more complicated topologies. Packet loss point detection can be performed across multiple network routes with a number of start and end points. For example, the problem analyst may be interested in performing packet loss detection from a given host to or from any of several workstations. In this case, the capture profile would specify the host as well as each workstation. A boolean expression can be used to indicate match criteria in the correlator application 500. In the present example, the host is monitored by probe P1 and the workstations are monitored by probes

P2 through P5. If the analyst is concerned only that the host packets arrive at any given workstation, then an expression such as (Seen at P1) and (Seen at P2 or P3 or P4 or P5) may be used to indicate a successful match. If a packet seen at P1 is not seen at any of P2 through P5 then an exception is generated. This function is also useful if the packet may take any one of several routes in traveling from a given host to a given workstation. If the problem analyst needs to verify that a packet sent from the host and intended for workstation W2 does in fact arrive at W2, then an expression such as (Seen at P1 and Destination is W2) and (Seen at P2) may be used. If a packet seen at P1 with a destination address of W2 is not also seen at P2, then an exception is generated. This approach may be used to perform analysis for an arbitrary number of network start and end points. Furthermore, the logs generated for a single capture interval by the collector can be analyzed in several different ways by providing the various boolean expressions to the correlator application 500. Thus, the problem analysts can analyze the log data and create exception data in a variety of manners in order to gain insight into the problem at hand.

[0029] In the embodiments of the invention discussed previously, the probes 106 and problem isolation system 108 operate in passive mode. This mode is a pure listening and collecting mode. In another embodiment of the present invention, the probes 106 and problem isolation system 108 operate in an active mode. In active mode, the probe 106 and problem isolation system 108 perform manipulations on network traffic. Thus, the data actually passes through the probe 106 and problem isolation system 108 back out to the network. As such, the probe 106 and problem isolation system 108 essentially insert themselves into the network link. The intent of such a function is to recreate a customer problem either in the customer environment or in a lab environment. Loss of packets can be simulated at various points in the network to generate host or workstation application failures. In addition, network congestion scenarios can be simulated by queuing packets in the probe capture buffers and releasing them at a prescribed rate. This will allow the

problem analysts to confirm a potential problem diagnosis and allow product owners to test the robustness of their products against a variety of network operating conditions.

[0030] As described above, the present invention can be embodied in the form of computer-implemented processes and apparatuses for practicing those processes. The present invention can also be embodied in the form of computer program code containing instructions embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other computer-readable medium, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. The present invention can also be embodied in the form of computer program code, for example, whether stored in a storage medium, loaded into and/or executed by a computer or transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via electromagnetic radiation, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. When implemented on a general-purpose microprocessor, the computer program code segments configure the microprocessor to create specific logic circuits.

[0031] While the invention has been described with reference to exemplary embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted for elements thereof without departing from the scope of the invention. In addition, many modifications may be made to adapt a particular situation to the teachings of the invention without departing from the essential scope thereof. Therefore, it is intended that the invention not be limited to the particular embodiments for carrying out this invention, but that the invention will include all embodiments falling within the scope of the appended claims.